

Projet DRONE

Gestion de L'OS Embarqué et de l'environnement Graphique

Pierre-jean TEXIER

Ecole Supérieure des Technologies Electronique Informatique Infographie

13 Février 2014



Sommaire

- 1 Présentation du Projet
- 2 Segment SOL
- 3 Gestion de Projet
- 4 Droit
- 5 Réalisations
- 6 Bilan de LOT
- 7 Conclusion

Présentation du Projet



Projet de FIN d'étude
"Contexte Industriel" durant 6 mois
PROJET Drone Next GEN
2 Composants => 2 Equipes



Présentation du Segment SOL

- Présentation



Présentation du Segment SOL

- Présentation



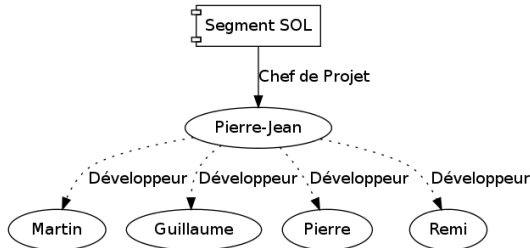
- L'équipe (OBS : **O**rganization **B**reakdown **S**tructure)

Présentation du Segment SOL

- Présentation



- L'équipe (OBS : **O**rganization **B**reakdown **S**tructure)

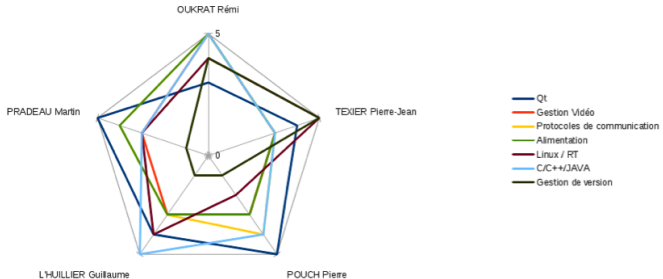


Expression des Besoins

Besoins Exprimés par le Client :

- Affichage
- Ergonomie
- Vidéo
- Communication
- Gamme de Température
 - Commerciale : 0°C à 70°C
 - Industrielle : -45°C à 85°C

Matrice de compétence Segment SOL

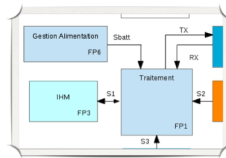


Remarques

Permet d'organiser au mieux les ressources

Cahier des Charges Personnel

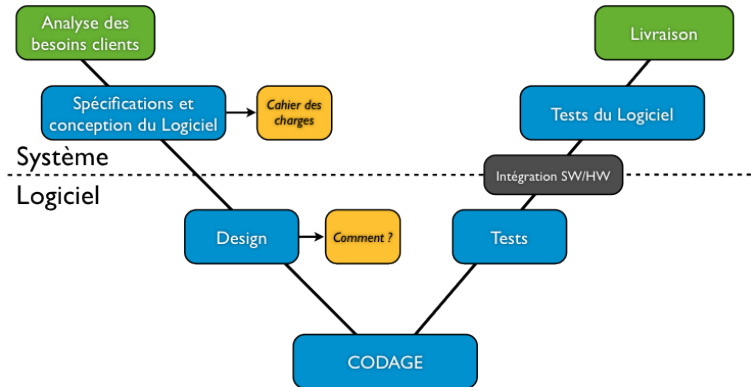
FP1 - FP6



Tâches à réaliser

- OS Linux embarqué Fonctionnel
- Préparation de l'environnement graphique (Qt, openCV, ...)
- Optimisation du temps de boot **hardware** et **subjectif**
- Gestion de l'énergie

Cycle de vie Logiciel



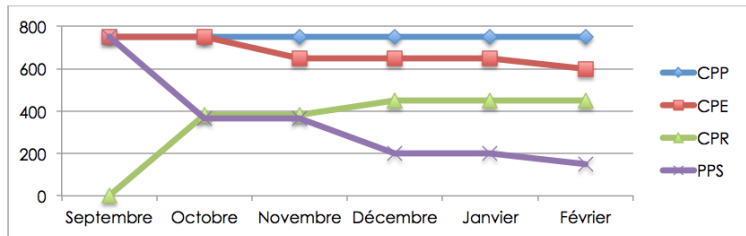
ROADMAP : Segment SOL

Phases	2013			2014		
	Sep.	Oct.	Nov.	Déc.	Jan.	Fév.
Analyse des besoins CLIENT / Cahier des Charges fonctionnelle du système.						
Définition architecture système et sous-système.						
Spécifications sous-systèmes et choix matérielles et logicielles.						
Conception et réalisation des composants spécifiques logicielles et matérielles.						
Tests unitaire du software, Qualifications des composants spécifiques hardwares.						
Test d'intégration des sous-ensembles et sous-systèmes						
Validation / Qualification du système complet.						

Remarques

- 2 Jalons : Intégration
- Suivi du cycle de Vie Logiciel

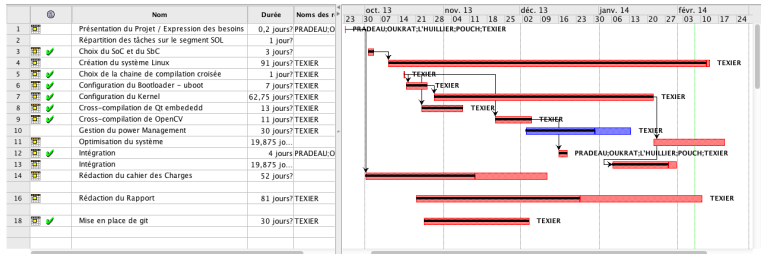
Suivi des dépenses : Segment SOL



Ce qu'il faut retenir

- 750 Euros de Budget à l'instant t0
- Environ 450 Euros dépensés en fin de PROJET

Gantt Prévisionnel



Remarques

Les tâches du cahier des charges sont présentées ainsi que les relations entre elles

Outils mis en place

Gestion de Version

- GIT

- ▶ [GITHUB Segment SOL](#)

Gestion de Documentation

- Doxygen

- ▶ [Doxygen Segment SOL](#)

Gestion de Publication

- Doku-Wiki

- ▶ [Wiki Segment SOL](#)

Droit

2 types de Licences utilisées pour le Projet

- GPLv3 

► Texte GPLv3

- Creative Commons 

► Texte Creative Commons

A savoir

- Développement avec des Outils Libres
- Améliore la **Maintenabilité**, **Portabilité** du développement

Quelques Outils

- GIMP, GNU Linux, GNU plot, bootchart, \LaTeX , fbvis, ...

Réalisations



Choix technologiques

Etude Materiel

- System On Chip ?

Allwinner A20 ▶ Synoptique

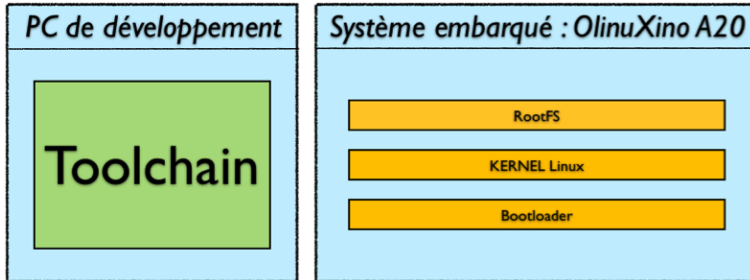
- 2 processeurs : Dual Core ARM Cortex A7
- 2 co-processeurs Graphique : GPU Mali 400MP2, ...

- Single Board Computer ?

OlinuXino A20 ▶ Disponible sur le marchés

- 160 GPIO
- Connecteur 40 Broches pour LCD, ...

Environnement “Linux Embarqué”



Chaine de compilation croisée

Compilation Croisée ?

Une chaîne de compilation croisée est un groupe d'outils permettant la compilation d'un programme d'une architecture processeur à une autre (x86 => ARM).

Chaîne de compilation croisée

Compilation Croisée ?

Une chaîne de compilation croisée est un groupe d'outils permettant la compilation d'un programme d'une architecture processeur à une autre (x86 => ARM).

Mon Choix



Chaîne de compilation croisée

Compilation Croisée ?

Une chaîne de compilation croisée est un groupe d'outils permettant la compilation d'un programme d'une architecture processeur à une autre (x86 => ARM).

Mon Choix



Possibilités

Crosstool-ng,



Chaine de compilation croisée

Compilation Croisée ?

Une chaîne de compilation croisée est un groupe d'outils permettant la compilation d'un programme d'une architecture processeur à une autre (x86 => ARM).

Mon Choix



Possibilités





Crosstool-ng,



Pourquoi HardFloat ?

- co-processeur : FPU neon-vfpv4

Risques et Opportunités

Composant	Risque			Opportunité		
Bootloader	Créer une image u-boot		Temps	Utiliser l'image présente sur la SD	Temps	
Kernel	Paramétrer/ Recompiler le Kernel	Optimisé pour le projet	Temps	Utiliser l'image présente sur la SD	Temps	Image trop lourde
RootFS	Créer		Temps	Utiliser le rootFS sur la SD	Temps	

Kernel in a Nutshell

Kernel ?

► Linux

Kernel in a Nutshell

Kernel ?

► Linux

Pourquoi Optimiser ?

Kernel in a Nutshell

Kernel ?

► Linux

Pourquoi Optimiser ?

- Empreinte Mémoire

Kernel in a Nutshell

Kernel ?

► Linux

Pourquoi Optimiser ?

- Empreinte Mémoire
- Besoins pour le Projet (V4L, Tactile, ...)

Kernel in a Nutshell

Kernel ?

► Linux

Pourquoi Optimiser ?

- Empreinte Mémoire
- Besoins pour le Projet (V4L, Tactile, ...)

Informations

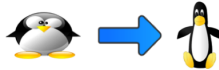
- Version 3.4.67
- Non mainline

► [Lien github](#)

A savoir

- Plusieurs branches
- 3.0 et 3.4.67 = stable
- 3.10 = experimental

Configuration



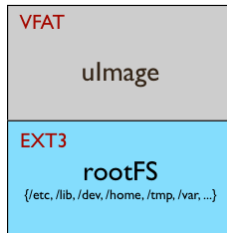
- Interface : xconfig (**make ARCH=arm xconfig**)
- Suppression :
 - Options inutiles dans l'embarqué (ex : Swap)
 - Options de Debug / Profilling (ex : Ftrace)
 - Options Inutiles pour notre Projet (ex : HDMI)

Empreinte Mémoire

Début du Projet : **5.20 MB** → Fin du Projet : **3.33 MB**

Implantation sur cible

- Système sur carte SD
- u-boot, image noyau et script.bin sur la 1ère partition
- rootFS sur la seconde partition



Préparation de l'environnement **GRAPHIQUE**

“Fournir l'abstraction bas niveau avec le développement Logiciel”

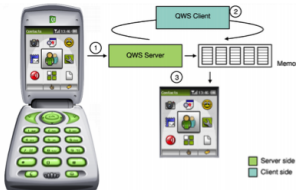
Qt embedded

Notre Contexte : 

Choix : Affichage graphique **Framebuffer**

Qt embedded ? :


- Framework IHM et application embarqué
- Linux - Android
- Architecture Serveur/Client : QWS



Qt embedded

Ce qu'il nous faut : “qmake” spécifique à notre architecture :

Etapes

- 1 Cross-compilation de la librairie Tactile : tslib 
- 2 Modification du fichier **qmake.conf**
- 3 Génération du Makefile spécifique aux besoins (./configure)
- 4 Cross-compilation de Qt embedded 4.8.2 (make)
- 5 Installation des binaires (make install)
- 6 Portage des binaires générés sur cible
- 7 Tests

OpenCV embedded



Portage sur Architecture ARM [▶ openCV](#)

Etapes

- 1 Choix des Modules openCV : Utilisation de Cmake
- 2 Cross-compilation librairies/modules
- 3 Portage des binaires générés sur cible
- 4 Tests

Optimisation du temps d'amorçage système



Hardware

U-boot

- Variable 'bootdelay'

Scripts de démarrage : ▶ bootchart

- Networking
- ssh
- exim4
- apache

Temps de BOOT

Début du Projet : **1 minute** → Fin du Projet : **33 secondes**

Subjectif

Remplacement du Logo de démarrage : *logo_linux_clut224.ppm*

- Logo de base



- Logo personnalisé : 640*480¹

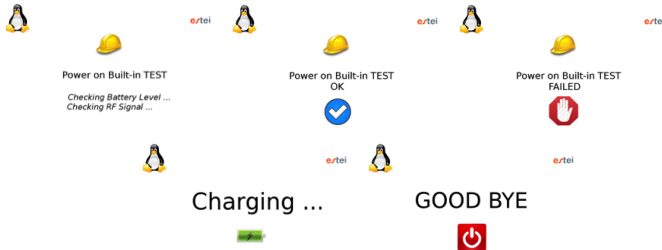


1. modifier script.fex

PBIT

Power On Built in Test

Utilisation simpliste du Framebuffer pour la phase de "PBIT"

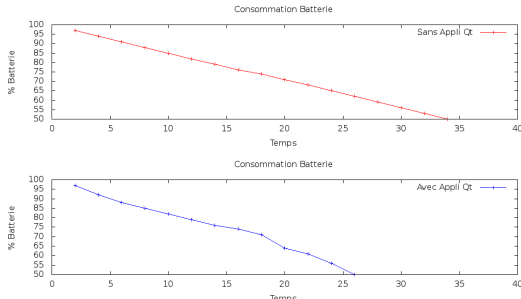


Cross-compilation de la librairie fbvis

Power Management

Réalisé

- Création d'un Crontab
- Test sur l'autonomie du Système



Optimisation du Système

Réalisé

- Système de fichier Temporaire
 - *tmpFS* (fichiers log)
- Autologin
 - *agetty –autologin*
- Lancement Automatique de l'application : ihm
- UDEV
 - iDVendor
 - iDProduct

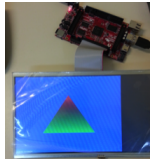
Optimisation du Système

Possible

Optimisation du Système

Possible

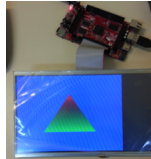
① Accélération matérielle



Optimisation du Système

Possible

① Accélération matérielle

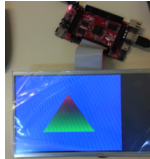


② Gestion des Heuristiques : Variations dynamiques de fréquence processeur

Optimisation du Système

Possible

① Accélération matérielle



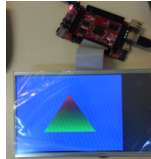
② Gestion des Heuristiques : Variations dynamiques de fréquence processeur

- powersave

Optimisation du Système

Possible

① Accélération matérielle



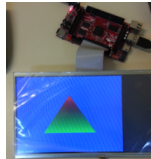
② Gestion des Heuristiques : Variations dynamiques de fréquence processeur

- powersave
- perfomance

Optimisation du Système

Possible

① Accélération matérielle



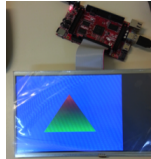
② Gestion des Heuristiques : Variations dynamiques de fréquence processeur

- powersave
- performance
- ondemand

Optimisation du Système

Possible

① Accélération matérielle



② Gestion des Heuristiques : Variations dynamiques de fréquence processeur

- powersave
- performance
- ondemand
- interactive, ...

LOT Segment SOL : Coûts

Coût de Developpement 

Nom / Prénom	Coût
TEXIER Pierre-jean	3300 €
PRADEAU Martin	2719 €
POUCH Pierre	2640 €
L'HUILLIER Guillaume	2640 €
OUKRAT Rémi	19 669 €

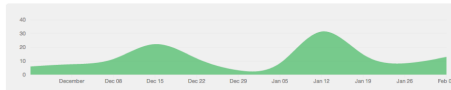
Coût D'industrialisation  : 100 Pièces => 33905 €

Conclusion






Lot Livrable



GIT : 249 Commits



Matrice de Validation


Cahier des Charges	TV	Commentaires
Choix SoC / SbC	Levée de risque sur carte	
Chaine de compilation croisée	Compilation "hello world"	
OS Linux sur cible	Sur carte SD	
Préparation graphique	Qt / OpenCV	
Power Management	Via sysFS	

Conclusion

Compétences Acquisées

- Portage d'application graphique sur *Architecture ARM*
- Optimisation d'un système Linux
- Gestion d'un Projet de bout en bout : *Chef de Projet*

Bilan Personnel

- Mise en pratique de l'enseignement 
- Orientation en *Linux Embarqué* confortée
- Atout pour le prochain stage
- Implication dans la communauté "SUNXI"

FIN

Tests de Validation

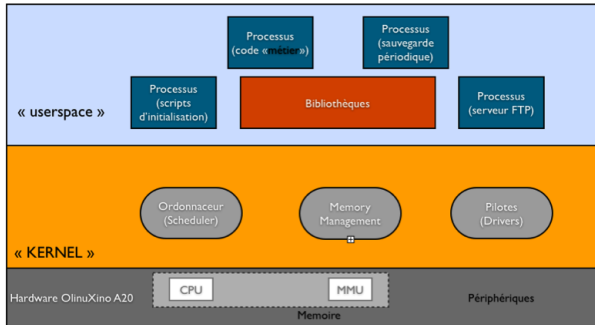
- 1 Démarrage
- 2 Version du noyau
- 3 Démo Qt embedded



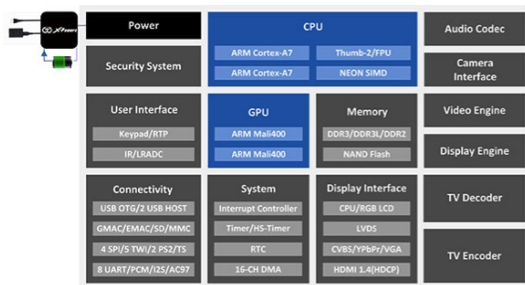
Questions



Système Linux



Synoptique du System On Chip



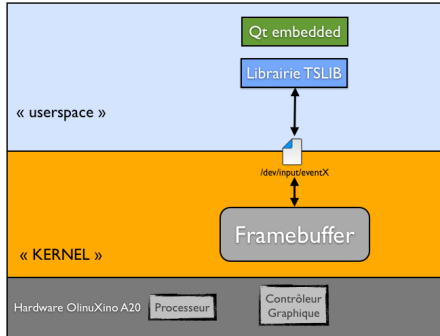
◀ retour

Etude du Marché

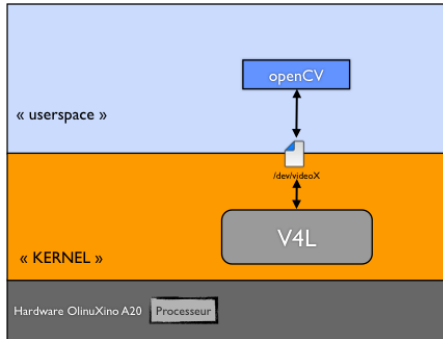


◀ retour

tslib



openCV



bootchart

